(12) 24 1

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) UNCLASSIFIED (12)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
2. GOVT ACCESSION NO. AD-A087624
3. RECIPIENT'S CATALOG NUMBER (9)

4. TITLE (and Subtitle) (6)
APPLICATIONS OF INFORMATION THEORY TO DATA COMMUNICATION NETWORKS.

5. TYPE OF REPORT & PERIOD COVERED
Paper, Technical rept.

6. PERFORMING ORG. REPORT NUMBER
(14) LIDS-P-1017

7. AUTHOR(a) (10)
Robert G. Gallager

8. CONTRACT OR GRANT NUMBER(a) (15)
ARPA Order No. 3045/5-7-75
N00014-75-C-1183
ARPA Order-3045

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Massachusetts Institute of Technology
Laboratory for Information and Decision Systems
Cambridge, Massachusetts 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Program Code No. 5T10
ONR Identifying No. 049-383

11. CONTROLLING OFFICE NAME AND ADDRESS
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

12. REPORT DATE (11)
July 1980

13. NUMBER OF PAGES
20

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Office of Naval Research
Information Systems Program
Code 437
Arlington, Virginia 22217

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

LEVEL II

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC ELECTE AUG 7 1980 S D C

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The use of information theory in the study of Data Communication Networks can be roughly divided into two parts. The first deals with the usual information theoretic problem of reliable and efficient transmission of information, generalized to treat multiple sources and destination. The second deals with the special problems caused by the bursty sources typical of data networks. These sources typically produce short messages interspersed with long and unpredictable intervals of silence. This makes the sources highly non-stationary over time

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

410950

80 8 4 247

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (continued).

intervals comparable to tolerable communication delays. This lecture will deal primarily with this second class of issues . We begin with a discussion of layering in networks and then proceed to a number of aspects of the data link control layer. We start with point to point links, discussing error control, framing, and generalized multiplexing, and then proceed to multi-point or broadcast links. Throughout, we focus on the information and source coding aspects of the required control.

Presented at NATO Advanced Study Institute, Univ. of East Anglia, Norwich, England, August 4-14, 1980.

July 1980 LIDS-P-1017

APPLICATIONS OF INFORMATION THEORY TO DATA COMMUNICATION NETWORKS*

Robert G. Gallager**

Laboratory for Information and Decision Systems,
Massachusetts Institute of Technology, Cambridge,
Massachusetts 02139 USA

ABSTRACT. The use of information theory in the study of Data Communication Networks can be roughly divided into two parts. The first deals with the usual information theoretic problem of reliable and efficient transmission of information, generalized to treat multiple sources and destination. The second deals with the special problems caused by the bursty sources typical of data networks. These sources typically produce short messages interspersed with long and unpredictable intervals of silence. This makes the sources highly non-stationary over time intervals comparable to tolerable communication delays. This lecture will deal primarily with this second class of issues. We begin with a discussion of layering in networks and then proceed to a number of aspects of the data link control layer. We start with point to point links, discussing error control, framing, and generalized multiplexing, and then proceed to multi-point or broadcast links. Throughout, we focus on the information and source coding aspects of the required control.

I. INTRODUCTION

One of the few things that are agreed upon by virtually everyone concerned with data networks is the need for layered

---

*This research was conducted at the M.I.T. Laboratory for Information and Decision Systems with partial support provided by grant DARPA/ONR-N000140650C-1183 and grant NSF/ECS-7919880.

**Present address.

design. Although the terminology used in this area is unfamiliar to many communication and information theorists, the concept has been so familiar to information theorists since the time of Shannon's original monograph that it is rarely even mentioned. We sometimes think of a point to point communication channel as a physical medium over which one transmits waveforms and, at the receiver, makes decisions on the transmitted data given the noise corrupted received waveform. Equally often, we think of the channel as containing the digital data modulator and demodulator (modem), and thus consider the channel as having a discrete (usually binary) input and output with conditional probabilities describing the outputs given the inputs.

The waveform channel is considered as the channel at one level and the discrete channel is at a higher level. For purposes of using the data, one may completely ignore the details of the physical channel and modem and simply treat the combination in terms of its input-output description (i.e., the discrete channel). Although the discrete channel is more convenient to work with than the waveform channel, it still contains a number of problems that are awkward for data networks. First, errors sometimes occur in the received symbols; second, the discrete channel is synchronous and one must transmit a symbol each transmission time whether or not there is something to send; and third, the sequence of symbols must carry both data for different users and all the control needed to distinguish the user data from the control and from idle periods.

The above awkward problems suggest creating a new type of higher level channel, which we shall call a frame channel. The input to the frame channel is a sequence of frames which are buffered at the input and accepted asynchronously; a frame is simply a finite sequence of binary digits. The output is the same sequence of frames as the input, transmitted without error but with variable delay. The frame channel is created from the discrete channel by putting data link control elements on each end of the discrete channel, just as the discrete channel was created from the waveform channel by putting modems on each end of the waveform channel. In brief, we have replaced a synchronous unreliable channel with an asynchronous reliable channel (see fig. 1).

One can go on and create yet higher levels of channels to replace either lower levels of channels or the entire network at a lower level; Zimmerman [1] gives a good description of the process involved and some international efforts at standardizing these levels. We shall confine our attention here to the frame level and to the requisite data link control. Despite the apparent simplicity of the topic there are a number of conceptually challenging problems buried here which appear to be good preparation for dealing with more general network and multiaccess problems.

We first treat point to point channels, dealing first with error control, then frame length, and then multiplexing of frames. We then turn to multiaccess channels.

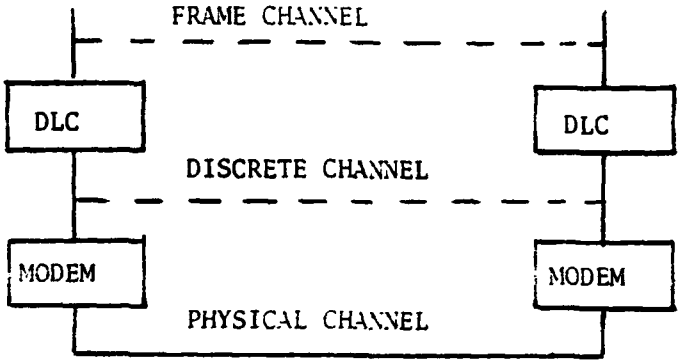

FIGURE 1

II. ERROR CONTROL

From a practical standpoint, there are two main approaches to error control--straight-forward error correction, and error detection with retransmission. Most communication channels, including channels leased from telephone networks, are corrupted not only by noise easily modeled as stationary random processes but also by bursts of noise of widely varying durations. Because of these bursts, it is not usually feasible to achieve the low error probabilities necessary for data networks by the use of error correction alone. Thus in most cases, error detection with retransmission is used either supplemented or not by error correction. From the standpoint of layering, it is appropriate to view error correction as part of the modem-namely one of the available techniques for synchronously transmitting bits reasonably reliably over a physical channel. Error detection and retransmission, however, is invariably regarded as part of the data link control because of its inherently asynchronous nature.

Aside from some outmoded error detection systems using horizontal and vertical parity checks, error detection is generally accomplished by cyclic redundancy checks (CRC); that is, the block or frame of binary data (regarded as a polynomial) is divided, in the modulo 2 field, by a check polynomial, and the remainder is appended to the data as check bits. International standard data link controls have settled on a particular polynomial, $x^{16} + x^{12} + x^5 + 1$, which is the product of a primitive

15^{th} degree polynomial and $x + 1$. For block lengths up to 2^{15}, this code will detect all bursts of at most 16 bits, all error patterns of at most 3 errors, and in the presence of completely random noise, will fail to detect errors with probability 2^{-16}. There are serious questions about whether this code is sufficiently powerful, especially since the error detectability improves rapidly with an increasing number of check digits. The essential point here is that with k check digits, the code detects all but a fraction 2^{-k} of the possible error patterns, and thus the error detecting capability increases rapidly with k and the particular choice of polynomial is not a primary consideration.* We shall return later to this question of error detectability, but now proceed to the question of how the transmitter knows when to retransmit erroneous frames.

The simplest type of retransmission strategy is the "stop and wait" strategy. A frame is first transmitted and then the transmitter waits until receiving either an acknowledgement (ACK) of correct reception or a notification (NAK) of incorrect reception. Then either the next frame is transmitted or the erronously received frame is retransmitted. The problem here is that the return channel is presumably as error-prone as the forward channel and thus error detection must also be used on the acks or naks. One solution here is to regard detected errors on the return channel (or no message at all) as a NAK and then to retransmit the old frame. This means, however, that the receiver could receive the same frame twice without knowing it was a repetition. This final problem could be resolved by numbering the frames modulo 2, thus allowing detection of repeated frames at the receiver. With these modifications, the strategy works correctly; by working correctly we mean that, given no failures in detecting errors on the forward and reverse channels, the frames entering the transmitter are delivered, in order and without repetition, to the receiver.

Stop and wait strategies are hopelessly inefficient, particularly on full duplex channels, since so much of the channel capacity is wasted in waiting. The so called "go back n" strategies, $n \geq 2$, avoid this problem by allowing the transmitter to continue transmitting new frames without waiting for acknowledgements. The restriction on going ahead is as follows: consider frames as being numbered sequentially as they enter the transmitter and let frame x be the lowest numbered frame that has not yet been acknowledged. The transmitter normally sends frames in order, except after transmitted frame $x + n - 1$, it next circles back and

---

*Leung and Hellman [2] have shown by counter-examples that one has to be slightly cautious with this argument.

transmits frame x again, followed by repeating x + 1, x + 2, and so forth.

Go back n strategies depend on how the transmitter and receiver inform each other of the frames being sent and acknowledged. There are a number of standard or well known data link controls such as HDLC, ADCCP, SDLC, BDLC, all of which are almost identical and all of which use a go back n strategy of the following type. The transmitted frame number, modulo 8, is sent within the frame itself. Also, since these strategies are inherently full duplex strategies, the acknowledgement information is carried within the data frames going in the opposite direction. In particular, each return frame carries the number, modulo 8, of the lowest numbered forward frame not yet correctly received. Each frame in each direction, then, carries 3 bits denoting the transmitted frame and 3 bits carrying acknowledgement information for the frames in the reverse direction. These 6 bits are carried in the second byte of the frame. When a frame is retransmitted, the contained acknowledgement information about reverse traffic is updated to the current value.

In analyzing go back n strategies, it is helpful to focus on transmission in one direction. We assume, in demonstrating that strategies work correctly, that the error detecting code never accepts a frame with errors. It is possible, however, because of the variable length frames, that the receiver will not even know that a frame with errors has been transmitted. Thus we consider each transmitted frame as being either received correctly or not received (either not detected at all or detected with errors). We assume that correctly received frames are received in the order transmitted, however. In the same way, acknowledgement messages in the reverse direction are either correctly received in the order sent, or not received.

For conceptual purposes, assume that the transmitter and receiver each keep track of the frame numbers being transmitted and accepted; each transmitted frame will contain the transmitted frame number modulo m (where m = 8 for HDLC). Let x, as a function of time, be the smallest frame number not yet correctly acknowledged at the transmitter and y be the smallest frame number not yet accepted at the receiver. It is clear that x and y are non-decreasing functions of time, and that at each time $x \leq y$. In order to precisely describe the algorithm at transmitter and receiver without presuming correct operation, let $\hat{x}$ and $\hat{y}$ be the estimate of x and y at transmitter and receiver respectively. Assume that initially $x = \hat{x} = y = \hat{y} = 1$, and, using induction on the time instants at which $\hat{x}$ and $\hat{y}$ are changed, assume that at a given time $x = \hat{x}$ and $y = \hat{y}$. We will show correctness then by showing that after any change, we still have $x = \hat{x}$ and $y = \hat{y}$.

In order to allow the transmitter to use any knowledge it might have about time delays to operate more efficiently, we generalize the transmitter strategy to allow it to transmit any frame with a number between $\hat{x}$ and $\hat{x} + n - 1$. We finally assume that $n < m$, and will soon see why this restriction is necessary. For the standard data link controls $m = 8$, this restricts go back n strategies to $n \leq 7$.

The receiver accepts a correctly received frame carrying the frame number k only if $k = \hat{y}$ modulo m; if k fails this test, the frame is rejected. Note that in this strategy the receiver does not accept frames out of order and then re-order them for delivery to the user; we will consider such an option later. Upon accepting a frame, the receiver increments $\hat{y}$. In each frame in the reverse direction the receiver inserts the current value of $\hat{y}$ modulo m as the acknowledgement number. When the transmitter correctly receives a frame in the reverse direction (whether or not that frame is "accepted" as above), it takes the acknowledgement number j and increases $\hat{x}$ by $(j-\hat{x})$ modulo m. It can be seen that the new $\hat{x}$ is either the value of $\hat{y}$ when the receiver sent that acknowledgement number or differs from that $\hat{y}$ be a multiple of m.

Now suppose that at a given time (with $x = \hat{x}$, $y = \hat{y}$), the receiver correctly receives frame number z. Using the x when that frame was sent, $x \leq z \leq x + n - 1$. Since $x \leq y$ at a given time and y is non-decreasing in time, $z \leq y + n - 1$ for the current value of y. On the other hand frame $y - 1$ has already been accepted by the receiver and was transmitted before the current frame z was transmitted. Thus $y - 1 \leq x + n - 1 \leq z + n - 1$. Combining these results,

$$y - n \leq z \leq y + n - 1 \tag{1}$$

It follows that if z mod $m = \hat{y}$, then since $y = \hat{y}$ and $m > n$, we must have $z = y$. Thus, if the receiver accepts z, we have shown that $z = y$, and the new incremented $\hat{y}$ equals the new y, establinghing correct operation at the receiver.

Next suppose the transmitter at a given time, with current $x = \hat{x}$, receives an acknowledgement number y modulo m, using the value of y when the acknowledgement was sent. Since the frames stay in order on the reverse channel, no larger value of y has been acknowledged at the transmitter previously, so $x \leq y$. Also, no frame has been sent with a number larger than $x + n - 1$, so

$$x \leq y \leq x + n - 1 \tag{2}$$

Since the new value of $\hat{x}$ can differ from the $y = \hat{y}$ only by a multiple of m, and $m > n$, the new value of $\hat{x} = y$, and this is also

the new value of x, establishing correct operation at the transmitter.

It is instructive to see what would go wrong with the above strategy if the receiver accepted frames out of order. Consider the following two scenarios. In the first, the transmitter sends frames 1 through n + 2 in order; frames n and n + 1 contain errors but the transmitter receives acknowledgements for frames 1 and 2 early enough to send n + 1 and n + 2. In the second the transmitter sends frames 1 through n followed by frame 1 since the acknowledgements do not get through to the transmitter. If m = n + 1, then the receiver has no way of distinguishing the two scenarios (especially if the received frame n in the second scenario looks like the received version of n and n + 1 in the first). Thus accepting this frame out of order for the first scenario leads to an error if in fact the second scenario had occured. This type of problem can be avoided if we choose $m \geq 2n$. Note that (1) is still valid if received frames are accepted out of order, and with $m \geq 2n$, the received frame can be uniquely identified from its modulus.

For strict go back n strategies where one retransmits successive frames after going back, there is little advantage to accepting frames out of order since those frames will be retransmitted anyway. For selective repeat request systems, in which the acknowledgement information actually identifies the erroneous frames, and only those frames are retransmitted, it is necessary to accept frames out of order. Such systems are desirable where the round trip delay is large, requiring a large value of n, and where the error probability is moderately large, causing serious inefficiencies when about n frames have to be retransmitted for each frame in error. We shall not go through the details of how to send the acknowledgement information for selective repeat systems, since the appropriate strategy is somewhat dependent on the details of the rest of the system. We hope, however, that the previous discussion of go back n strategies has alerted the reader to the somewhat treacherous subtleties involved in such systems.

We look next at the amount of control information involved in go back systems. While the 6 bits per frame actually used in the HDLC strategy is certainly not exhorbitant, it is at least of academic interest to see how much of this is necessary. Assume for simplicity that the transmission delay is known in both the forward and reverse direction. The only acknowledgement information useful to the transmitter then is a record of which transmitted frames are correctly received. Assuming that there is a probability p that a frame is not correctly received, the uncertainty at the transmitter about the sequence of correct receptions at the receiver is

$$H(p) = - p \log p - (1-p) \log (1-p) \quad (3)$$

This uncertainty is only reduced by dependence of correct reception on the past and on frame length. Since H(p) is at most one bit, the useful information in the acknowledgements sent to the transmitter is at most one bit per frame.

Next, making the reasonable assumption that the receiver knows the transmitter's strategy, the only uncertainty at the receiver about which frame number is being transmitted is its uncertainty about which acknowledgements have been received. Assuming the acknowledgements are imbedded in the reverse frames, this uncertainty is again at most H(p') where p' is the probability that a frame in the reverse direction is not correctly received. Since the information needed to identify the frame number in one direction is the same as that for acknowledgements in the reverse direction, we see that H(p) upper bounds the entire required control information in one direction and H(p') in the other. We shall return later to describe a method for greatly reducing the number of binary digits required for the control of retransmission. Before that, we treat the problem of specifying the lengths of messages or of frames.

III. MESSAGE LENGTHS AND FLAGS

We start off by viewing the information inside the frames of an error detection and retransmission system as simply a stream of binary digits, where in principle there need be no relation between frame boundaries and the boundaries of individual messages. The error detection guarantees that this stream can be transmitted without errors, but if the stream is a concatenation of individual messages, there is no guarantee that the receiver can separate the stream into these messages. For example, suppose that the set of possible messages is 0, 1, 00, 01, 10, 11 with the first two occuring with probability 1/4 each and the last four with probability 1/8 each. The entropy of the message set is 5/2 bits, but the average message length is 3/2 binary digits. The problem is that the binary digits in the message do not carry all the information in the message; one bit is carried in the length of the message. Thus it is reasonable (and almost optimal) to encode the messages by first encoding the message length into a source code and following the source code word by the message digits themselves (see [5] for a more complete discussion). The receiver then decodes the length, then reads out the message digits, and then is ready to decode the length of the next message.

If the message lengths are geometrically distributed with mean length M, then the entropy of the length distribution approaches log e M = log M + 1.44 for large M (all logs here are

base 2). The Huffman code for this distribution is equivalent [3] to choosing an integer $L \simeq M/\log e$ and then breaking the message into 0 or more packets of length L and a final packet of length 1 to L. Each non-final packet is preceded by a 1 and the final packet is preceded by a 0 followed by an encoding of its length.

A more interesting type of encoding is generated by viewing the sequence of messages as a sequence of bits separated by markers denoting the ends of messages. Thus the sequence is viewed as a sequence of ternary symbols, 0, 1, and marker, where the 0 and 1 are equally likely and the marker is relatively improbable for a large mean message length M. The flag strategy is a simple and highly efficient way of encoding such a ternary sequence into binary digits (far more efficient than the single letter Huffman code). The strategy is to encode 0 into 0, 1 into 1, and the marker into a sequence of r bits called a flag. Naturally a problem arises if the flag sequence appears within a message. This problem is circumvented as follows: whenever r-1 consecutive bits of a message match the first r-1 bits of the flag, then an insertion is made after these r-1 bits, the insertion being the complement of the final flag bit. At the receiver, after seeing the first r-1 bits of the flag, the next bit is deleted if it is the complement of the final flag bit, and otherwise the flag is recognized. Note that if the transmitter only inserted a bit after r-1 matches when the r th bit matched also, then the receiver could not tell when bits should be deleted.

The mean number of binary digits used to represent the length of a message in this strategy is r (since each message is terminated by a flag) plus the mean number of insertions per message. For messages of a given length $m \geq r - 1$, the probability of an insertion after bit i, $r-1 \leq i \leq m$, is 2^{-r+1}, so $(m-r+2)\ 2^{-r+1}$ is the mean number of insertions in messages of length $m \geq r - 1$. For $m < r - 1$, there are no insertions. Upper bounding $m - r + 2$ by m and averaging over message lengths, the mean number of insertions per message is at most $M\ 2^{-r+1}$. Choosing r as one plus the integer part of log M, we see that the mean number of binary digits required to represent the length of a message by the flag strategies is at most 2 + log M. For a geometric length distribution, this exceeds the entropy by only .56 binary digits, but the bound of 2 + log M is valid for any length distribution with mean M. It is rather astonishing that such a simple source coding technique is so efficient.

The flag strategy has a wide variety of potential applications in network protocols. We first describe how it is used in the standard data link controls such as HDLC and then describe a possible application for increasing the efficiency of go back n

retransmission systems.

The structure of a frame in HDLC is Address, Control, Data, CRC, Flag. The address is 8 bits long and is used for multiplexing different users. The control is 8 bits long also and contains the frame numbers for forward and reverse traffic as described before. The data field is of variable length and might even be empty, which is indicated by one of the additional bits in the control field. The CRC is the 16 bit check sequence described earlier and checks the address, control, and data fields. Finally, the flag is the fixed 8 bit sequence 0111 1110; a flag must also precede each frame, but the flag terminating one frame can also serve to precede the next. If there are no frames to send, the transmitter just sends repeated flags. In terms of our previous description, only the first seven bits of this flag are used to denote the end of a frame, and insertions are used within the frame whenever the first six bits 011111 appear. The eighth bit of the flag is used to distinguish the flag from a rarely used control character consisting of a zero and seven ones. In order for a frame to be correctly received, it is necessary both for the CRC checks to be satisfied and for the flag to appear after the CRC; in fact, the receiver only knows where to look for the CRC after it receives the flag.

One peculiar feature of this strategy is that a single error in a frame could create a flag inside the frame, and if the previous 16 bits happened to satisfy the CRC checks, an undetectable error would occur. It is highly unlikely for such an unusual pattern of data bits to occur, but this data dependent feature of the error detection scheme is unfortunate. There are only two known ways of correcting this problem and neither is very attractive from a system viewpoint. One is to use fixed length frames and the other is to encode the length of one frame into a field of the previous frame (which would eliminate the need for flags). Another peculiar feature of the strategy is that either an error in a flag or a flag created by errors causes a loss of frame synchronization; this is the primary reason why retransmission strategies cannot rely on knowing how many frames in error they have received.

Another feature of HDLC is that it is intended for applications where each frame carries data from only one message, but a message can be divided into several frames. One of the eight positions in the control field is used to denote whether or not a frame is the final frame of a message. Thus the combination of this bit and the flags encode the message lengths. One should not, however, conclude that frame lengths should be chosen to maximize the information theoretic efficiency of this encoding. The major tradeoffs that effect the choice of frame length are as

follows: first, long frames yield a high efficiency in terms of the ratio of data bits to control bits (there are 40 control bits per frame in HDLC, counting address, control, CRC, and flag). On the other hand, on relatively noisy channels, long frames can lead to an excessive number of retransmissions, thus lowering efficiency. For the more common case of relatively noise free channels, long frames lead to excessive delay in going through successive nodes in a network, since the entire frame is generally checked for errors at a node before allowing transmission on the next channel toward the destination. It is possible to send frames through intermediate nodes without checking, but this is not as simple as it appears.

We now describe how to use flags in a go back n error detection and retransmission system. The general idea is that if errors are unlikely, then the event of the transmitter going back n frames is a rare event, which can be efficiently encoded by a flag. Similarly the event of not acknoledging a frame will be a rare event which can also be encoded by a flag. What this means is that frames are normally acknowledged implicitly. When a frame with no flag is correctly received in the reverse direction, the transmitter assumes that all frames in the forward direction were correctly received up to some past point in time. This past time is the time at which the receiver could have inserted a flag minus the time to send a maximum length frame (since it can take the receiver that long to recognize that an error has occured). Naturally the transmitter has to take the round trip propagation into account too, so this scheme presumes a known upper bound for round trip propagation and processing and a maximum frame length.

In its simplest form, the receiver follows the following strategy: it has two states, good and bad. In the good state (subject to a slight proviso) it accepts correctly received frames as they arrive and inserts no repeat request flags in outgoing frames. When an error is detected, or a maximum length frame comes in with no end of frame detection, the receiver goes into the bad state. In the bad state it accepts no frames and places a repeat request flag in each outgoing frame with the number, modulo m, of the lowest numbered frame it hasn't yet accepted. When a frame containing a retransmit flag is correctly received, the receiver goes back into the good state. The retransmit flag is accompanied by the number, modulo m, of the transmitted frame. This number might not agree with the frame awaited by the receiver and the proviso referred to above is that the receiver just counts correctly received incoming frames without accepting them until the desired frame arrives.

The transmitter normally transmits frames in order without retransmission flags and uses the reverse frame to update its count x of the lowest numbered unacknowledged frame. If a frame

arrives on the reverse channel with a repeat request flag, the transmitter updates x to agree with the requested frame and then sends this frame with a retransmission flag and the frame number modulo m. Also, after sending frame number x + n - 1, the transmitter goes back to x, accompanying it with a retransmission flag and frame number.

It can be seen that as p (the probability that a frame is not correctly received) goes to zero, the probability of sending flags and frame numbers goes to 0, although there are still occasional insertions to avoid matches between the data in the beginning of a frame and the flag. It can be seen that a single error, in the strategy described, leads to several repeat request flags and several retransmission flags. This can be avoided by complicating the strategy. Other techniques for reducing retransmission control information are given by Golestaani [4].

IV. MULTIPLEXING AND ADDRESSING

A communication channel in a data network typically carries messages from a large number of sources directed to a large number of destinations. We call the sequence of messages from a given source to a given destination a conversation. Since these messages arrive randomly in time and are typically separated by large time intervals, it is necessary to somehow identify which message is associated with which conversation on each communication link. Conceptually, we associate each conversation with an address which is an encoding of the source and destination.

There are two standard and well known ways of identifying messages with conversations on a communication link. The first is time division multiplexing (TDM) (or frequency multiplexing at the physical channel level) and the second is statistical time division multiplexing (STDM) [5],[6]. In TDM, one divides the bit stream on the channel into slots, consisting of a fixed number of bits each, and then one cycles through the set of conversations using the channel, associating one slot to each conversation (or perhaps several slots in a cycle to one conversation if the conversations have different rates). There is no need to transmit addresses in this scheme since transmitter and receiver can both identify conversations by positions in the cycle. Such a system is very efficient under very heavy loading because of the absence of addressing overhead (although there is the need to reserve a special slot sequence to indicate absence of any data). Such a system leads to very long delays, however, because of the failure to dynamically share the communication resources between the users; for this reason TDM is rarely used for data (as opposed to voice conversations).

In STDM, one allows variable length slots, thus increasing

efficiency when a conversation has little data to send, and one avoids slots with no data by instead sending the address of a conversation along with the data. Naturally one needs to encode the length of each slot as discussed above in section 3. STDM is highly effective for lightly loaded channels, since if only one conversation has data to send at a given time, a slot can be allocated immediately. STDM is less effective for heavily loaded channels because of the addressing overhead and is also less effective for conversations producing information at a steady rate.

The standard data link controls use STDM with each frame corresponding to a slot. The address is contained in the first eight bits of the frame and we have previously discussed the encoding for frame lengths. The Tymnet network [7] uses STDM in a different way, where each frame for error detection purposes consists of a multiplexing of slots, each addressed and each containing a variable number of data characters for the addressed conversation.

Since TDM has illustrated that addresses are not really needed to identify conversations, it is reasonable to ask whether there are intermediate positions between TDM and STDM. While this may appear to be an academic pre-occupation with addressing efficiency, we shall find more practical import for these questions in multi-access systems. The first idea that comes to mind is to combine the cycling of TDM (which avoids the need for addresses) with the variable slot lengths of STDM. Whether or not the conversation associated with a slot has data to send can be encoded by a single binary digit, 0 representing no data, 1 representing data. If there is data, the 1 representing data to send can be followed by an encoding of the number of binary digits to be sent followed by the data itself. Thus the bit stream

1, length a, data a, 0001, length b, data b,...

would indicate that the first conversation had length a bits of data, given by data a, the second to fourth conversations had no data, and the fifth conversation had lengths b bits given by data b. It is equivalent to regard the slots as only the slots with data, with each slot preceded by a unary encoding of the number of intervening conversations in the cycle with no data.

If there are n conversations using the channel, then STDM requires log n address bits, whereas the above scheme uses a number which fluctuates with the loading. Under very heavy loading, with most conversations having something to send, the address information is usually one binary digit, whereas with very light loading, the address would have a mean length of about n/2. It is seen from this that the unary encoding of conversations in the cycle with nothing to send is very inefficient for light loading.

If we assume that the number of conversations around the cycle with nothing to send is geometrically distributed, then we recall that we have already discussed the Huffman code for geometrically distributed integers. It fragments the set of conversations into blocks of L conversations each, and uses a unary code for the number of blocks with nothing to send, followed by the address within a block of the next conversation with data to send; L is ideally given by the mean number of conversations with no data, divided by log e, but processing is simpler if L is a power of 2. Choosing L = 1 yields the original unary code, which is highly efficient under heavy loading, and choosing L = n is equivalent to the addressing of STDM. Intermediate values of L are of course more appropriate for intermediate levels of loading.

The above strategies all seem somewhat ad hoc, and it seems more fundamental to ask how much information (information theoretically) is required to identify messages with conversations on a channel. If one cycles between serving the different conversations, then it is perfectly reasonable to ask about the entropy of the number of intervening conversations with no data. On the other hand, if one serves messages in a first come first served order, then the entropy is very different.

A more fundamental approach to the identification of messages comes from the recognition that the arrival times of messages contain information. As a somewhat facetious example, when one's child calls from college, the occurence of the message means that the child needs money; there is often no additional information in the body of the message. On a communication channel, with multiplexing of other messages from other sources, there is generally a variable delay in the time required for sending the message, so the receiver gets an estimate (with some distortion) of the arrival time of the message. Whether or not the receiver is interested in this information is immaterial; the point is that it has been transmitted and thus has taken up part of the capacity of the channel. In a TDM system, this information is contained in the indications whether or not slots contain data; under very heavy loading, when almost all slots contain data, this information is very small, but the delay is large and highly variable, so the receiver gets very little arrival time information. In STDM, the arrival time information is contained in the addresses, whereas in the generalized STDM strategies just discussed, the arrival time information is in the encoding of how many conversations in the cycle contain no data.

The analysis of the amount of information about message arrival times contained in delay distorted estimates of those times is carried out in [8], where the problem is formulated as a rate-distortion problem. It is shown that if the message arrivals for a conversation are Poisson with rate α and if the expected delay

before delivery at the receiver is d, then the information per message supplied to the receiver about message arrival times is at least

$$- \log (1 - e^{\alpha d})$$

It is also shown in [8] that in the limit of a very large number of multiplexed conversations, a strategy essentially equivalent to the generalized STDM strategy here uses at most .6 bits per message for addressing more than the above information theoretic limit.

The assumption of a very large number of conversations being multiplexed together has the effect of allowing one to ignore queueing delays. In [9], several strategies somewhat more sophisticated than the generalized STDM strategy here were analyzed and compared, taking queueing delays into account. Under very heavy loading, it was shown that delay is surprisingly sensitive to the particular strategy used.

V. MULTIPLE ACCESS CHANNELS

Consider a class of channels containing one central station and a set of remote stations. Data transmitted from the central station is received by each of the remote stations and data transmitted from each remote station is received by the central station. If several remote stations transmit at the same time, the central station detects that transmission is taking place but can not correctly receive the data. We assume operation at the frame level, so that frames are correctly received so long as the remote stations transmit one at a time. Such a model can be applied to multidrop telephone lines, a satellite with multiple ground stations, a radio channel with one central station, etc.

We discuss two types of systems, generalized polling systems and generalized slotted Aloha type systems. Our major theme will be to relate these systems to the addressing ideas presented in the last section. We start with polling because the connection is particularly close there. Polling is appropriately regarded as a sequence of binary questions and answers, the questions being posed by the central station and the answers provided by the remote stations. What we shall see is that if we regard each remote station as generating one conversation, then the binary digits used for addressing in the last section can be regarded as answers to polling questions.

First consider conventional polling with remote stations numbered 0 to n-1. The central station first addresses station 0, asking if it has a message. Station 0 responds with a yes or no answer, and sends the message if the answer is yes. Station 1 is

next polled in the same way and so forth, cycling around the set of stations. This conventional polling corresponds to the unary coding policy discussed in the last section. The single binary digit used there to indicate whether a conversation has a message corresponds to the binary response to the poll of the corresponding remote station. The inefficiency of the unary code for light loading corresponds to the inefficiency of conventional polling for lightly loaded systems, but the problem is more serious with polling because of the large time for the poll and response.

Next, consider generalized polling, or probing, as developed by Hayes [10]. The idea is that if only one of n stations contains a message, it should be possible to identify that station with fewer than n questions; in fact, information theory indicates that log n binary questions should be sufficient. Suppose for example that there are 16 remote stations and each station is represented by its own 4 bit binary address. Suppose that station 0110 is the only station with a message to send. Suppose further that the central station first asks if any remote station has a message and that stations with messages respond by a short burst of carrier on the channel. The central station detects from the response that one or more stations have messages and next asks if any station with an address starting with 0 has something to transmit. The answer (from station 0110) is again yes, so the central station next asks if a station with address starting with 00 has something to transmit. The answer is no, since no carrier appears on the channel. The central station concludes from this that all remote stations with messages have addresses starting with 01. The central station next asks about remote stations with addresses starting with 010 (this is a slight modification of [10], in which the 01 stations are first polled; we eliminate this poll since the answer is already known to be yes). The answer to the 010 question is no and the central station concludes that one or more stations with 011 addresses have messages. The next question asks if 0110 has a message. After answering affirmatively, station 0110 sends its message.

If we encode yes answers as 0 and no answers as 1, we see that the sequence of answers for the above example is 00110. The sequence of the last four binary digits is 0110, the address of the identified station. Thus, aside from the initial 0, the question answers are the same as the address used in STDM. If we think a little more about STDM, however, we realize it is necessary to distinguish idle periods from transmitted messages, and the initial zero simply corresponds to that choice.

Next suppose the above example is modified so that both station 0110 and station 1011 have messages. The generalized polling proceeds as before until the transmission of the message

from 0110. The central station then asks if there are any other stations with messages. The answer is yes, so the stations starting with 0 are queried. The answer is no, and 10 stations are queried with a yes answer. Next the 100 stations are queried and then the station 1010 is queried. The answer is no, it is concluded that 1011 has a message and the station is commanded to send its message. The entire set of question answers is 0-0110-0-1011-1, where the final question is if any other stations have something to send. One should be convinced at this point that the answers in the generalized polling system correspond exactly to the addresses in STDM.

We saw before that STDM was somewhat wasteful of addressing bits for intermediate ranges of loading and that the mean number of addressing bits could be reduced by partitioning the conversations into blocks, using a unary code to cycle through the blocks containing messages and then identifying the conversation within a block by an address. The same strategy can be used with polling. The central station first asks if the first block has messages. If the answer is yes, the central station identifies the remote station(s) by successively refined questions as before and then proceeds to the next block. Again the answers to the queries correspond, one to one, with the addressing bits used in the addressing strategy. This polling strategy (except for the occasional redundant question) was also devised by Hayes [10], who also suggested that the strategy could be made adaptive to loading conditions. The association of polling questions with addressing, however, is original here.

It is reasonable to ask whether any addressing strategy can be matched with a generalized polling strategy, and the answer is no. An addressing strategy is really an encoding of the set of conversations with messages into binary code words. Such an encoding can only be associated with a polling strategy if each binary digit in the code tree can be interpreted as the answer to a question if there are messages in some set of conversations. Fortunately, however, the best encodings that have been discovered have this property.

Conceptually, it is interesting to observe that in a polling system, it is not necessary for the central station to ask questions. The remote stations could figure out the questions for themselves if they knew the answers to the previous questions. Thus the central station could simply acknowledge the answers to the questions to the remote stations rather than asking questions, and this brings us to slotted Aloha type systems.

In slotted Aloha type systems, we regard time as being segmented into slots, each slot being long enough for the transmis-

sion of one frame of data. The remote stations are synchronized to this slot time so that two remote stations either transmit frames simultaneously and neither are correctly received, or they are completely separated in time and are correctly received. In the original system [11],[12], a message arriving at a remote station is transmitted in the next slot time. If a conflict occurs (two or more stations transmit simultaneously), each message is retransmitted at a later randomly chosen slot. This system has some stability problems in the sense that if too many stations get in the retransmit mode, the probability of a frame getting through correctly drops almost to zero. A more sophisticated strategy for resolving conflicts, called the binary tree algorithm, was devised and analyzed by Capentenakis [13] and independently discovered by Hayes [10]. For later improvements see [14],[15].

The idea of the binary tree algorithm is as follows: if a conflict occurs, then on the next slot, all stations with addresses starting with 0 transmit if they had messages involved in the conflict, and on the following slot, stations with addresses starting with 1 transmit. If conflicts occur in either of these slots, the corresponding set of stations is further divided into sets specified by the first two bits of the address, and so forth until the messages are correctly transmitted. New messages coming in are delayed until after the conflicts are resolved.

It can be seen that this strategy is clearly related to the STDM form of generalized polling just described. The initial transmission by all stations with messages corresponds to answering the question whether any station has a message. Further transmissions correspond to answering questions about subsets of remote stations. One difference is that the central station does not ask questions, but rather informs the remote stations of the answers, thus allowing the remote stations to follow the algorithm. The other difference is that if only one frame is transmitted in a slot, the remote station is identified by the correctly received message, and it is not necessary to use further slots to identify the station. Thus we lose the precise identification between slots and the binary digits in a code to identify messages.

This strategy is appropriately modified under heavier loading to cycle through subsets of stations rather than having all stations initially transmit. In fact, as in generalized polling, it is best to do this modification dynamically, based on the duration of the previous conflict resolution interval.

REFERENCES

[1] H. Zimmerman, "OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection", IEEE Trans. Comm., Vol. COM-28, pp. 425-432, April 1980.

[2] S.K. Leung and M. Hellman, "Concerning a Bound on Undetected Error Probability", IEEE Trans. I.T., Vol. IT-22, pp. 235-237, March 1976.

[3] R.J. Camrass and R.G. Gallager, "Encoding Message Lengths for Data Transmission", IEEE Trans. I.T., Vol. IT-24, pp. 495-497, July 1978.

[4] S.J. Golestaani, "Design of a Retransmission Strategy for Error Control in Data Communication Networks", Report ESL-R-674, Electronic Systems Laboratory, M.I.T., Cambridge, Mass., July 1976.

[5] W.W. Chu, "A Study of Asynchronous TDM for Time-Sharing Computer Systems", AFIPS Conference Proceedings, Fall Joint Computer Conference, pp. 669-678, 1969.

[6] P. Baran, et al, "On Distributed Communications", Rand Corp, Santa Monica, Calif., August 1964.

[7] L. Tymes, "TYMNET--A Terminal Oriented Communication Network", AFIPS Conference Proceedings, Spring Joint Computer Conference, pp. 211-216, 1971.

[8] R.G. Gallager, "Basic Limits on Protocol Information in Data Communication Networks", IEEE Trans. I.T., Vol. IT-22, pp. 385-398, July 1976.

[9] P.A. Humblet, "Source Coding for Communication Concentrators", Report ESL-R-798, Electronic Systems Laboratory, M.I.T. Cambridge, Mass., January 1978.

[10] J.F. Hayes, "An Adaptive Technique for Local Distribution", IEEE Trans. Comm., Vol. COM-26, pp. 1178-1186, August 1978; also Bell Laboratories TM-76-3116-1, March 1976.

[11] N. Abramson, "The Aloha System", in Computer Communication Networks, N. Abramson and F. Kuo, Eds., Englewood Cliffs, N.J.: Prentice-Hall, 1973.

[12] L.G. Roberts, "ALOHA Packet System With and Without Slots and Capture", Comput. Commun. Rev., Vol. 5, pp. 28-42, April 1975.

[13] J.I. Capetanakis, "Tree Algorithm for Packet Broadcast Channels", IEEE Trans. I.T., Vol. IT-25, pp. 505-151, September 1979; also Report ESL-R-806, Electronic Systems Laboratory, M.I.T., Cambridge, March 1978.

[14] R.G. Gallager, "Conflict Resolution in Random Access Broadcast Networks", Proceedings AFOSR Workshop in Communication Theory and Applications, Sept. 17-20, 1978, Provincetown, Mass, pp. 74-76.

[15] J. Mosely, "An Efficient Contention Resolution Algorithm for Multiple Access Channels", M.S. Thesis, M.I.T., Dept. of Elec. Eng. & Comp. Sc., May 1979.

Distribution List

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 Copies
Assistant Chief for Technology Office of Naval Research, Code 200 Arlington, Virginia 22217	1 Copy
Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217	2 Copies
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, Massachusetts 02210	1 Copy
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, Illinois 60605	1 Copy
Office of Naval Research Branch Office, Pasadena 1030 East Greet Street Pasadena, California 91106	1 Copy
New York Area Office (ONR) 715 Broadway - 5th Floor New York, New York 10003	1 Copy
Naval Research Laboratory Technical Information Division, Code 2627 Washington, D.C. 20375	6 Copies
Dr. A. L. Slafkosky Scientific Advisor Commandant of the Marine Corps (Code RD-1) Washington, D.C. 20380	1 Copy

Office of Naval Research
Code 455
Arlington, Virginia 22217

1 Copy

Office of Naval Research
Code 458
Arlington, Virginia 22217

1 Copy

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego, California 92152

1 Copy

Mr. E. H. Gleissner
Naval Ship Research & Development Center
Computation and Mathematics Department
Bethesda, Maryland 20084

1 Copy

Captain Grace M. Hopper
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350

1 Copy

Advanced Research Projects Agency
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

1 Copy

Dr. Stuart L. Brodsky
Office of Naval Research
Code 432
Arlington, Virginia 22217

1 Copy

Captain Richard L. Martin, USN
Commanding Officer
USS Francis Marion (LPA-249)
FPO New York 09501

1 Copy